Online Services

[News] [Real-Time Software Support (RTSS)]

MVME5500 Early Access VxWorks BSP

Contents

Description

This release is the 0.4 version of the BSP for the Motorola MVME5500 board on the Tornado 2.2.1 release. It contains a complete copy of the Motorola MVME5500 BSP. The target.nr file has been updated to include configuration specific information essential to the operation of the MVME5500 BSP. Please review the BSP documentation before using the BSP.

Items Supported and NOT Supported

Supported

- MPC7455/MPC7457 Processor
  - This support includes any errata work-arounds required.
- L1, L2, and L3 Cache
- L3 private memory
- MPx bus mode
- MMU (Memory Management Unit)
- Decrementer driver
- 512GB to 1GB SDRAM with ECC
- SPD (Serial Presence Detect) SROMs
- GT-64260 ASIC System Controller
  - Host PCI bridge functions
  - Memory controller
  - I2C interface
  - Device bus
  - Interrupt controller
  - DMA driver
  - Timer/Counter as Auxiliary Clock
- Tundra Universe 2
  - PCI-VME bridge
  - DMA
  - VxWorks Shared Memory
- Extended VME memory model
- VPD (Vital Product Data)
- Flash memory
  - Up to 64MB soldered (flash0)
  - 8MB socketed flash (flash1)
  - Boot image in either bank
- MK48T37
  - 32KB NVRAM RTC
  - Alarm Clock
  - Failsafe Timer
- Serial Console (16550) Driver
- GT-64260 Integrated Ethernet Driver
  - (wancom)
- Wind River Gigabit Ethernet Driver
  - (gei)
- PCI Auto Configuration
  - PMC Module Configuriation
- IPMC module
  - SCSI
  - Serial
- PMCspan
- Onboard fail LED

NOT Supported

- Abort Switch
- TFFS (True Flash File System)
- VME
  - DMA controller in linked list mode

Contents of BSP

The BSP tar image that you downloaded is named `MVME5500_0.4.tar.gz`. Extract this file into a temporary area. It contains the following files:

```
README.MVME5500
```

```
MVME5500_BSP0.4.tar
```

The `MVME5500_BSP0.4.tar` file contains the following directories and/or files:

```
target/config/mv5500/
Motorola.lic
WRS.lic
```

The first item is the directory for the MVME5500 BSP from Motorola. Motorola.lic and WRS.lic are the license files that govern this release of the BSP.

Prerequisites

The following items must be installed BEFORE installing the BSP:

Wind River's "Tornado 2.2.1/VxWorks 5.5.1 for PowerPC"
Wind River's "BSPs/Drivers for VxWorks 5.5.1: PowerPC"

Installation Instructions

The BSP is packaged as a "tar file within a tar file". The outer tar file contains two files: a README.MVME5500 file with instructions for installing the BSP, and another tar file with the BSP itself. Winzip can be used to extract the files from the downloaded tar file on a Windows 95/98 or Windows NT system.

Installation Instructions for Unix

1. Install Tornado 2.2.1 support from Wind River.
2. Make sure the appropriate environment variables are set according to the VxWorks Tornado User's Guide.
3. Install the files contained in this BSP by performing the following:
    a. Change to your VxWorks root directory.
    If you view the directory you will see the "`target`" directory among others.
    2. Uncompress and untar the downloaded file.

```
gunzip MVME5500_0.4.tar.gz
tar -xvf MVME5500_0.4.tar
tar -xvf MVME5500_BSP0.4.tar
```

4. Read the file `target/config/mv5500/README` for initial BSP information.
5. Rebuild the bootrom and the kernel (instructions in "target.nr"). Note that a "boot.bin", "vxWorks.st" and "vxWorks" binary image are included as part of the BSP. These binaries have been built with the default "config.h" and can be immediately used on your board.
6. Make sure you FLASH the bootrom binary before using the kernel.

Installation Instructions for Microsoft Windows 95/98/NT

1. Make sure the Tornado package from Wind River is installed.
2. Open the winzip application.
3. Use the "OPEN" winzip action to find and open the downloaded file.
4. Once the downloaded tar file is opened, the Winzip window will display the names of the README and BSP tar files.
5. To view the README file contents, select it and choose the "VIEW" winzip action.
6. To extract the files, select the README and BSP tar files and execute the "EXTRACT" winzip action. You will be prompted to choose a directory into which the extracted files will be stored. This should be set to the Tornado installation directory (default is c:\Tornado; if you view the directory you will see the "target" directory among others). After providing the directory information, choose the "Extract" option in the "Extract" window. The extracted files will be written into the directory. (Note: WinZip may be so fast that it seems nothing happened.)
7. Use the "OPEN" winzip action to find and open the BSP tar file. This will be in the Tornado installation directory that you previously entered.
8. Once the tar file is opened, the Winzip window will display the names of the files which are contained in the archive. On the right side of the list, the destination path for each file will be displayed.
9. To install the BSP, select the files and execute the "EXTRACT" winzip action. You will be prompted to choose a directory into which the extracted files will be stored. This should be set to the Tornado installation directory. After providing the directory information, choose the "Extract" option in the "Extract" window. The extracted files will be written into the destination directories.

Known Limitations

None.

Additional Information

1. Socketed vs. Soldered ROMs

   The J8 jumper governs which flash-ROM runs, socketed (flash1) or soldered (flash0). MOTLoad has been programmed into the socketed (flash1) flash-ROM and a vxWorks bootrom ("boot.bin") can be programmed into the soldered (flash0) flash-ROM. See the reference pages below under "ROM considerations" for an explanation of how to do this. The board is shipped with the J8 jumper set to enable MOTLoad.

2. How to Flash the Bootrom

   An explanation of how to do this is contained in the reference pages below in the section titled: "Flashing the Boot ROM Using Motorola MOTLoad".

MVME5500

NAME

MVME5500 - Motorola MVME5500

INTRODUCTION

This manual entry provides board-specific information necessary to run VxWorks. Before using a board with VxWorks, verify that the board runs in the factory configuration by using vendor-supplied ROMs and jumper settings and checking the RS-232 connection. This BSP is compatible with Wind River's Tornado 2.2.1 development environment.

This BSP encompasses the MVME5500 Single Board Computer. The MVME5500 board is a VMEbus single-slot Computer based on the PowerPC MPC7455/MPC7457 processor with integrated L1 and L2 cache as well as backside L3 cache, and the Galileo GT-64260B host bridge with dual PCI Interface and Memory Controller. On-board payload includes two PMC slots, two SDRAM banks (up to 1GB of memory), an expansion connector for two additional banks of SDRAM (for an additional 1GB of memory), 8MB of socketed Boot FLASH ROM (also referred to as "flash1" or "bank B"), up to 64MB of on-board soldered flash (also referred to as "flash0" or "bank A"), one 10/100/1000 Ethernet port, one 10/100 Ethernet port, two RS232 serial ports, and 32KB NVRAM/Real-Time Clock/Failsafe Timer and Universe 2.0 PCI to VME bridge. The MVME5500 supports the IPMC761 PMC card for MVME761 I/O functionality and the IPMC712 PMC card for MVME712 I/O functionality.

Boot ROMS

The MVME5500 supports two banks of FLASH memory. Flash0 (32 MBytes) consists of two Intel StrataFlash 3.3 volt devices configured to operate in 16-bit mode and is soldered onboard. Flash1 consists of four 56-pin TSSOP sockets which will be populated with 8MB of Flash memory using 16-bit wide Intel StrataFlash devices. Either bank can be used as the boot bank, which will be jumper selectable. The jumper effectively swaps the chip selects (GT-64260) of the two FLASH banks. When the jumper is placed on pins 1 and 2 bank A becomes the boot bank. When the jumper is placed on pins 2 and 3 bank B becomes the boot bank. Bank B contains Motorola's MOTLoad firmware.

Boot Line Parameters
To load VxWorks, and for more information, follow the instructions in the Tornado User's Guide: Getting Started.

Jumpers

The following jumpers are relevant to VxWorks configuration:

Jumper	Function	Description
J8	Boot ROM controller	Across pins 1 and 2 to select the soldered FLASH (flash0). Across pins 2 and 3 to select the socketed FLASH (flash1).
J27	System controller selection	Across pins 1 and 2 for no SCON. Across pins 2 and 3 for auto-SCON. Not installed for "always SCON".
J28	PMC/IPMC	Across pins 1 and 2 for PMC mode. Across pins 2 and 3 for IPMC761 mode. Across pins 1 and 2 for IPMC712 mode.
J32	PMC/IPMC	Across pins 1 and 2 for PMC mode. Across pins 2 and 3 for IPMC761 mode. Across pins 2 and 3 for IPMC712 mode.
J19	Bus mode select	Across pins 1 and 2 for 60x mode Across pins 2 and 3 for MPX

		mode
J17	SROM initialization	Across pins 1 and 2 for GT-64260 SROM initialization. Across pins 2 and 3 for no GT-64260 SROM initialization.
J6,J100,J7,J101	10/100 Ethernet	Across pins 1 and 2 to route 10/100 ethernet to front panel. Across pins 2 and 3 to route 10/100 ethernet to MVME761.
J97,J34,J98,J99	10/100 Ethernet	Leave pins 1 and 2 unjumpered to route 10/100 ethernet to front panel. Across pins 1 and 2 to route 10/100 ethernet to MVME761.

For jumper configuration details, see the board diagram at the end of this entry and in the hardware manual.

FEATURES

The following subsections list all supported and unsupported features, as well as any feature interaction.

Supported Features

The following features of the MVME5500 board family are supported:

Feature	Description
Processors	MPC7455/MPC7457 Up to 100MHz bus clock L1, L2 and L3 cache(including Private Memory) MMU (Memory Management Unit) Decrementer
DRAM	Up to 2GB SDRAM SPD (Serial Presence Detect) SROMs ECC (Error Checking and Correcting)
GT-64260	Host Bridge functions Memory Controller I2C interface Interrupt controller Device bus DMA functionality Internal timer as Auxiliary Clock
Bus Mode	MPX and 60x bus mode is supported
Tundra Universe 2	PCI-VME bridge VME Interface: 32-bit address, 32-bit data PCI bus interface A32/A24/A16, D32/D16/D08 master and slave programmable interrupter and interrupt handler full system controller function eight location monitor/signal registers DMA controller (in direct mode only).
VPD	Vital Product Data
FLASH	Up to 64MB Flash (flash0 or BANK A, 16-bit) 8MB Flash (flash1 or BANK B) Boot image in either bank
MK48T37	32KB NVRAM Real-time clock Alarm Clock Failsafe Timer
Peripherals	Two async serial debug ports; One 10/100 Mb Ethernet interface; One 10/100/1000 MB Ethernet interface;
PCI Interface	64-bit PCI/PCI-X; complies with PCI Local Bus Specification, Revision 2.1
Fail LED	Extinguish after board initialization
IPMC761 Interface	Two additional asynchronous serial ports. Two additional asynchronous/synchronous serial ports. SCSI interface.

Unsupported Features

The following features of the MVME5500 board family are not supported:

Feature	Description
VME Interface	DMA controller in linked list mode (only direct mode is supported).

```
      TFFS          True File Flash System for the Intel StrataFlash
      Miscellaneous ABORT switch

Feature Interactions

     None known.

HARDWARE DETAILS

     This section details device drivers and board hardware elements.

     Memory ECC Protection
          This BSP supports ECC memory which is enabled by default. To
          disable ECC, #undef INCLUDE_ECC in config.h.

Devices

     The device drivers, libraries and support routines included with
     this BSP are:

      byteNvRam:            byte-oriented generic non-volatile RAM driver.
      i8250Sio:             Intel 8250 UART driver (debug port).
      m48t37:               M48T37 Timekeeper SRAM device driver.
      sysMv64260AuxClk:     GT-64260 auxiliary clock driver.
      sysMv64260Dma:        GT-64260 DMA driver.
      sysMv64260I2c:        GT-64260 I2C serial EEPROM driver.
      sysMv64260Int:        GT-64260 interrupt controller driver.
      sysMv64260Phb:        GT-64260 Host Bridge support.
      pciAutoConfigLib:     PCI auto-configuration library.
      pciConfigLib:         PCI configuration library.
      pciConfigShow:        Show routines of PCI bus library.
      ppcDecTimer:          PowerPC decrementer timer driver (system
                            clock).
      Smc:                  GT-64260 System Memory Controller support.
      sysMv64260SmcShow:    System Memory Controller configuration Show
                            routine.
      sysCache:             MPC745x L1, L2 and L3 cache support.
      sysFailSafe:          STMicroelectronics Watchdog/Failsafe Timer
                            driver.
      sysMotVpd:            Vital Product Data support.
      sysMotVpdShow:        Vital Product Data Show routines.
      sysRtc:               Real-Time clock and alarm clock support
                            routines.
      universe:             Tundra Universe chip VME-to-PCI interface
                            driver.
      wancomEnd.obj:        10baseT/100baseTX GT-64260 ethernet driver.
      ns8730xSuperIo(IPMC): Super I/O controller driver.
      z8530Sio(IPMC):       Zilog 8536 UART driver.
      sym895Lib(IPMC):      SYM53C895A SCSI driver.

BSP CONFIGURATION

     Most BSP configuration values are taken from on-board Vital
     Product Data (VPD) and Serial Presence Detect (SPD) serial
     EEPROMs. If invalid SPD is encountered (as determined by incorrect
     checksum), the memory controller is programmed with default values
     allowing access to 256MB of DRAM. Executing sysMv64260SpdShow( )
     at the VxWorks prompt will reveal if the SPD data is valid or not.

PCI Dynamic Allocation Spaces

     PCIx_MSTR_IO_SIZE, PCIx_MSTR_MEMIO_SIZE, PCIx_MSTR_MEM_SIZE, and
     ISA_MSTR_IO_SIZE (where "x" is "0" or "1" relating to GT-64260 bus
     0.0 and 1.0) control the sizes of the available PCI address
     spaces. There is one set of definitions for each bus. The windows
     defined by these parameters must be large enough to accommodate
     all of the PCI memory and I/O space requests found during PCI
     autoconfiguration. If they are not, some devices will not be
     autoconfigured. These definitions can be found in config.h.

      NOTE: PCI auto-configuration is performed by the bootroms. Any
            changes to
            PCIx_MSTR_IO_SIZE, PCIx_MSTR_MEMIO_SIZE, PCIx_MSTR_MEM_SIZE, or
            ISA_MSTR_IO_SIZE (where "x" is "0" or "1") requires the
            creation of a new
            bootrom image.

Memory Maps

     Diagrams of the CPU-PCI, PCI-VME and VME-PCI memory mapping are
     presented in config.h.
```

On-board RAM always appears at address 0x00000000 locally.

Dynamic memory sizing is supported. By default, LOCAL_MEM_AUTOSIZE is defined so memory is auto-sized at hardware initialization time. If auto-sizing is not selected, LOCAL_MEM_SIZE must be set to the actual size of DRAM memory available on the board to ensure all memory is available. The default fixed RAM size is set to 32MB (see LOCAL_MEM_SIZE in config.h).

Note that LOCAL_MEM_SIZE only controls the amount of memory mapped by the MMU. It does not control the amount of memory detected and configured by the Bootrom. The amount of physical memory indicated by the Serial Presence Detect data determines the memory controller configuration and, if enabled, the ECC initialization range.

L3 Cache Private Memory Support

The BSP provides support for L3 cache private memory. This support is not defined by default. To include private memory support change #undef INCLUDE_CACHE_L3_PM to #define INCLUDE_CACHE_L3_PM in config.h.

The private memory can be configured and initialized with the routine sysL3CachePmEnable(size) from application code, or from the kernel prompt. The size parameter indicates the desired private memory size, 0x100000(1MB) and 0x200000(2MB) are the only valid parameters. Note that the private memory must not be enabled and initialized until the MMU has been initialized.

The private memory support can be disabled by calling the routine sysL3CachePmDisable( ).

Also, the routines sysL3CacheFlushDisable( ) and sysL3CacheInvEnable( ) have been designed to preserve the private memory configuration.

Shared Memory

On all boards, shared memory across the backplane can also be used as a network interface. The name of the shared memory is sm. The BSP can be configured for shared memory support by #define'ing INCLUDE_SM_NET in config.h.

Shared memory network communications requires a signaling method and a method of mutually exclusive memory resource access. Signaling can be done using software polling or interrupts. By default, mailbox interrupts are used and SM_INT_TYPE is set to SM_INT_MAILBOX_1. To use polling, #define SM_INT_TYPE as SM_INT_NONE.

There are master and slave windows into VME address space to access the VME mailbox registers so that each CPU can send and receive shared memory interrupts using single-byte mailboxes. The windows map a 4KB region in A32 space at address 0xFB000000 + (0x1000 * CPU #) into the Universe chip registers. This configuration allows one processor to generate a SIG1 interrupt in another processor by accessing the other processor's mailbox register and setting the SIG1 bit. Each CPU has a master window covering the A32 addresses 0xFB000000 through 0xFB00ffff representing CPU numbers 0 through 15. Each CPU's slave window maps the appropriate address for that CPU to the Universe chip's register set.

Note that the Universe II location monitors are no longer used for shared memory interrupt notification. All four Universe II location monitors are available for user applications and have been left in their default state (disabled and unmapped).

Shared memory resource mutual exclusion (spin lock) is implemented using test-and-set (TAS) and clear operations on byte-sized semaphores. The #define for SM_TAS_TYPE is set to SM_TAS_HARD inside of "mv5500A.h", a requirement if VxMP is used. Hardware TAS and clear operations are performed by the sysBusTas( ) and sysBusTasClear( ) routines, respectively, and invoke pseudo-atomic operations.

True atomic operations are those which cannot be preempted at the hardware level and appear on a bus as a single-cycle instruction. Pseudo-atomic operations are composed of multiple instruction cycles executed on a bus that is locked (owned) by the processor executing the instructions. This is the method used when the

```
MVME5500 hosts the shared memory pool (SM_OFF_BOARD set to FALSE).

The routine sysBusTas( ) performs pseudo-atomic TAS operations by
disabling interrupts (to prevent deadlocks) and locking ownership
of the VMEbus. This routine waits up to 10 microseconds to lock
the bus. If bus ownership has not been achieved at the end of this
period, the routine returns FALSE, the same as it would if the
semaphore had already been set.

The GT-642260 on the MVME5500 is not capable of translating a VME
RMW bus cycle into an onboard atomic operation. Thus if the
MVME5500 is used in a shared memory configuration and is compiled
with SM_OFF_BOARD set to FALSE (indicating that the shared memory
pool is onboard), then ANY_BRDS_IN_CHASSIS_NOT_RMW must be
#define'd.

When using the MVME5500 in a shared memory configuration with
another type of VME board such as the MVME5100. The following
slight modification must be made to the last few lines of the
MVME5100's sysBusTas( ) function:

#endif /* ANY_BRDS_IN_CHASSIS_NOT_RMW */
        }
    else    /* Slave node */
        {

#if 0
        /* A board with the UNIVERSE_II can generate a VMEbus RMW */

        return (sysVmeRmwTas(adrs));
#else
return (sysVmeVownTas(adrs));
#endif
        }
    }

The effect is to invoke sysVmeVownTas( ) instead of
sysVmeRmwTas( ).

Interrupts

The system interrupt vector table has 256 entries. Vectors numbers
are grouped according to the following table:

 Vector# Assigned to
 ------------------------------------------------------
 0x00 - 0x1f GT-64260 Main Cause (low) interrupts
 0x20 - 0x3f GT-64260 Main Cause (high) interrupts
 0x40 - 0x4f GT-64260 GPP interrupts
 0x50 - 0x5f Universe related interrupts.
 0x60 - 0xbf Reserved - currently unassigned.
 0xc0 - 0xff Available for user application software.

As the above table shows, the BSP uses interrupt vectors (numbers)
beginning with 0 and proceeding to numerically higher values.
Interrupts such as VME bus interrupts which can be assigned a
value by the user should be confined to numbers in the range 0xc0
thru 0xff to avoid conflict with system required interrupts. It is
suggested that application software avoid the use of 0xff as an
interrupt vector. Although a legal number, 0xff often is
associated a nonresponding PCI read and may cause confusion when
debugging.

The interrupt number is numerically equivalent to interrupt
vector.

The GT-64260 interrupt controller does not directly support
interrupt priorities. Software configuration can be performed
however, which affects the order in which interrupts sources are
checked upon occurrence of an external processor interrupt. The
#define for ICI_MAIN_INT_PRIORITIES (in config.h) specifies the
order in which GT-64260 main interrupt cause bits are checked.

In addition to the interrupts associated with the GT-64260 main
interrupt cause register, the GT-64260 contains a 32-bit multi
purpose port (MPP). The MPP pins can be configured as external
interrupt pins through association with the General Purpose Pins
(GPP). In addition some of the MPP pins are configured for control
and status purposes. The table below summarizes the MPP pin
assignments.

 Pin# Vector# I/O  Polarity  Source
```

0	0x40	I	High	COM0/COM1
1	0x41	I	High	Unused
2	0x42	I	Low	Abort interrupt
3	0x43	I	Low	RTC & thermostat interrupts (ORed)
4	NA	O	Low	Unused
5	NA	O	Low	Unused
6	0x46	I	Low	Watchdog WDNMI# interrupt
7	0x47	I	Low	LXT971A interrupt (10/100Mbit PHY)
8	0x48	I	Low	PMC 1 INTA#
9	0x49	I	Low	PMC 1 INTB#
10	0x4a	I	Low	PMC 1 INTC#
11	0x4b	I	Low	PMC 1 INTD#/IPMC INT
12	0x4c	I	Low	Universe/VME interrupt VLINT0
13	0x4d	I	Low	Universe/VME interrupt VLINT1
14	0x4e	I	Low	Universe/VME interrupt VLINT2
15	0x4f	I	Low	Universe/VME interrupt VLINT3
16	0x50	I	Low	PMC 2 INTA#
17	0x51	I	Low	PMC 2 INTB#
18	0x52	I	Low	PMC 2 INTC#
19	0x53	I	Low	PMC 2 INTD#
20	0x54	I	Low	Intel 82544 interrupt
21	0x55	I	Low	Unused
22	0x56	I	Low	Unused
23	0x57	I	Low	Unused
24	0x58	I	Low	Watchdog WDNMI#
25	0x59	I	Low	Watchdog WDE#
26	0x5a	O	High	GT-64260A SROM init active output
27	0x5b	I	Low	Unused
28	0x5c	O	Low	Unused
29	0x5d	O	Low	Unused
30	0x5e	I	Low	Unused
31	0x5f	I	Low	Unused

For further details, refer to the appropriate board's reference guide.

Each "interrupt" Pin# from the above table is associated with a bit in the local GPP interrupt cause register. In addition, one of the four "main interrupt cause low" bits (24, 25, 26, or 27) is set indicating that a GPP related interrupt has occurred. When one of these four main interrupt cause bits indicates a GPP interrupt, the interrupt handler will expect to see one or more of the interrupt bits from the above GPP set to be active. The order in which the GPP interrupt set is checked is governed by the #define GPP_LOCAL_INT_PRIORITIES which is found in config.h.

PCI Auto-Configuration

To simplify the addition of PCI-based add-in cards, the BSP provides a PCI auto-configuration library.

The auto-configuration is called from sysHwInit to discover and configure the installed PCI devices and bridges. PCI auto-configuration is only called one time - during bootrom initialization. Device configuration includes the following PCI information:

Base Address Registers (BARs)

Space in the address map is dynamically allocated to each valid BAR detected. Allocation pools are maintained for the following PCI address spaces:

16-Bit PCI I/O

32-Bit PCI I/O

PCI Memory I/O (non-prefetchable memory)

PCI Memory (pre-fetchable memory)

Interrupt Routing

The correct interrupt vector number is placed in the intLine register of the device's PCI header. To connect to the device's interrupt, use the VxWorks intConnect( ) function with the value read from intLine.

PCI Header Completion

The PCI auto-configuration library fills in the remainder of the PCI header as follows:

Cache Line Size = _CACHE_ALIGN_SIZE/4

```
          Latency Timer = PCI_LAT_TIMER

          Command Register = I/O enabled, Memory enabled and Bus Master
          enabled.
```

IPMC Devices

Note that the hardware and BSP must match in regards to the IPMC module. Do not attempt to define IPMC features without installing an IPMC module. IPMC features should be defined when an IPMC is installed. Jumpers J28 and J32 must be configured for the proper IPMC mode as well.

Serial Configuration

The MVME5500's two asynchronous serial interfaces are provided by the TL16C550C Universal Asynchronous Receiver/Transmitters (UARTs) interfaced to the GT-64260 Device Bus. COM0 is routed to a front-panel RJ45 connector. COM1 is interfaced to an on-board 9-pin connector. An optional IPMC transition module will add four additional interfaces available through the rear panel transition board. By default COM2 and COM3 are defined as the Super I/O serial ports and COM4 and COM5 are the Z8536/Z85230 serial ports.

By default, the serial port is configured as asynchronous, 9600 baud, with 1 start bit, 8 data bits, 1 stop bit, no parity, and no hardware or software handshake. Hardware handshake using RTS/CTS is a supported option.

SCSI Configuration

SCSI is implemented via the IPMC module. To include SCSI support change #undef INCLUDE_IPMC, in config.h, to #define INCLUDE_IPMC.

In order for the SYM53C895A to perform to its full potential - of wide ultra2 SCSI - the driver needs to be configured using the routine scsiTargetOptionsSet( ). The following is an example showing how to configure the driver, the code would be added to sysScsiConfig( ) in sysScsi.c:

```
    scsiId = SCSI_SET_OPT_ALL_TARGETS;
    which = ( SCSI_SET_OPT_XFER_PARAMS | SCSI_SET_OPT_WIDE_PARAMS);

    options.minPeriod = 6;
    options.maxOffset = 255;
    options.xferWidth = 1;

    if (scsiTargetOptionsSet (pSysScsiCtrl, scsiId, &options, which)
        == ERROR)
        {
        printf ("Could not set target option parameters\n");
        return (ERROR);
        }
```

In order for sysScsiConfig( ) to execute during initialization #define SCSI_AUTO_CONFIG, in config.h, must be changed to #undef SCSI_AUTO_CONFIG, and #undef SYS_SCSI_CONFIG must be changed to #define SYS_SCSI_CONFIG.

GT-64260 DMA Configuration

To enable DMA support using the GT-64260, change the #undef INCLUDE_MV64260_DMA in config.h to #define. The functions specified in sysDma.h and sysDma.c should be used to develop applications that are DMA device independent. Specific details about the GT-64260 DMA functions are contained in mv64260Dma.h and sysMv64260Dma.c. All eight DMA channels are supported in both direct mode and chain mode. The header files "mv64260Dma.h" and "sysDma.h" contain #define's which may be useful to application code which uses the DMA functions. It is suggested that these files be #include'd by the application code.

GT-64260 DMA snooping is turned off by default to achieve the optimal DMA speed. Be aware that turning off IDMA snooping will result in disabling snoop transactions even in the cache coherency regions. Define IDMA_SNOOP_ON in config.h if snoop transactions are desired.

NOTE

When using IDMA chain mode, be sure that the chained descriptors

are stored in Little Endian mode in memory. The VxWorks LONGSWAP macro can be used to facilitate the creation of Little Endian descriptors.

Universe/VME DMA Support

DMA support is implemented as a synchronous "VxWorks driver", that is the calling task will be blocked in an interruptable polling loop until the DMA transfer has terminated. To keep this driver as simple as possible, only direct-mode operations will be implemented, that is, linked-list mode will not be supported.

This driver is strictly non-sharable; however, it contains no guards to prevent multiple tasks from calling it simultaneously. It assumes that the application layer will provide atomic access to this driver through the use of a semaphore or similar guards.

As a precaution, it is recommended by the Tundra User's Manual that the calling task set up a background timer to prevent an infinite wait caused by a system problem. Also, tasks transferring large blocks of data should lower their priority level to allow other tasks to run, and tasks transferring small blocks of data should use bcopy( ) instead of calling this driver.

Network Configuration

The GT-64260 provides one 10/100 full duplex ethernet port. A second gigabit ethernet port is provided by the Intel 82544 controller. Each port is accessed by front panel RJ45 connectors. INCLUDE_GEI_END is #define'd by default, causing the Intel 82544 gigabit ethernet port (gei0) to be instantiated and initialized. To make the GT-64260 ethernet port available, INCLUDE_WANCOM_END must be changed from #undef to #define.

If INCLUDE_NETWORK, INCLUDE_END, and INCLUDE_WANCOM_END are all #define'd then "wancomEnd.obj" must be included as part of the MACH_EXTRA list in the Makefile (remove the "#" in front of "wancomEnd.obj"). If any of the above are not #define'd, then "wancomEnd.obj" must be removed from the MACH_EXTRA list. Failure to follow this will result in build errors.

The Ethernet driver automatically senses and configures the port as 10baseT, 100baseT, or 1000baseT (if appropriate).

VME Access

VME access windows are documented in the memory map inside of config.h.

VMEbus accesses can be classified as either master or slave. A master access is one in which the accessing processor has bus mastership (it owns the bus) and is addressing resources on another VME board (the slave board). The master addresses the off-board resources through a memory mapping mechanism which assigns portions of the local address space to the various VME address spaces. These local memory regions are windows onto the VMEbus. Each window is individually configured with a set of base addresses -- one for the local bus, the other for the VMEbus -- and a window size.

A slave access is one in which slave VME processors allow access to their resources from the various VME address spaces through slave windows.

The normal VxWorks default is to enable the slave access windows only on the VxWorks node which is configured as processor zero in the bootrom parameter list. Otherwise, slave accesses are normally not permitted.

The default configuration for processor number zero maps all local memory onto VME A32. There are no A24 or A16 slave windows.

There is no support for the A64/D64 VME extensions.

To disable any VME master or slave window, just set the appropriate VME_Axx_xxx_SIZE macro (in config.h) to 0.

NOTE

Only the macros in config.h are considered user options. Macros in mv5500A.h or mv5500B.h should not be changed by the user.

Failsafe Timer

NOTE

The RTC (Real-Time clock) must be running in order to use the VxWorks sysFailsafe functions. If the timer is not running it can be set using the "set" command inside of MOTLoad. See the MOTLoad "help set" output to determine the syntax of this command.

Support for a failsafe (i.e. watchdog) timer is provided. The failsafe timer is implemented with the M48T37Y Timekeeper SRAM. This support is not part of the standard VxWorks watchdog library, wdLib. Failsafe timer expiration can be reported via a maskable interrupt or via a board reset event. The timeout lengths range from 0 (disable) to 31 seconds.

Failsafe timer support can be included in the BSP by defining INCLUDE_FAILSAFE in config.h. This support by default is excluded. There is only one failsafe timer on the board, so only one failsafe timer can be established at any given time.

The failsafe timer is disabled at power-up and after a reset. The failsafe timer support routines are defined in sysFailsafe.c.

In order to use the failsafe timer, the user will need to first call sysFailsafeSet( ). The routine takes as parameters the number of seconds until expiration and whether or not to generate a board reset upon expiration. If reset is set to FALSE, an interrupt occurs, if reset is set to TRUE, a board reset occurs. Passing a value of 0 for seconds will disable the failsafe timer. Once the timer has been set, subsequent calls to sysFailsafeSet( ) will extend the timer for the specified number of seconds.

A call to sysFailsafeCausedReset( ) will determine whether the failsafe timer caused the last board reset. This information will be lost if a call to sysAlarmSet( ) is made prior to calling sysFailsafeCausedReset( ).

A call to sysFailsafeCancel( ) will disable the failsafe timer. The current failsafe timer settings can be retrieved with a call to sysFailsafeGet( ). The current failsafe timer settings can be displayed with a call to sysFailsafeShow( ), this displays the current settings not the number of seconds until timer expiration. The routine sysFailsafeIntr( ) is the failsafe timer interrupt handler. In order to define your own interrupt handler, simply edit this routine.

Real-Time Clock and Alarm Clock

Support for a real-time clock and an alarm clock are provided. The real-time and alarm clocks are implemented with the M48T37Y Timekeeper SRAM.

Real-time and alarm clock support are included in the BSP by defining INCLUDE_RTC in config.h. This support by default is excluded.

When the real-time clock information matches the alarm clock settings an interrupt will be generated.

Once set, the alarm clock will retain its settings upon a board reset. The real-time and alarm clock support routines are defined in sysRtc.c.

The real-time clock can be set with a call to sysRtcSet( ). The following information needs to be supplied in order to set the RTC: century, year, month, day of month, day of week, hour, minute, and second. The current RTC settings can be retrieved with a call to sysRtcGet( ). The current RTC date and time can be displayed with a call to sysRtcShow( ). The sysRtcDateTimeHook( ) routine is provided as a hook to the vxWorks dosFsLib as a means of providing the date and time for file timestamps.

The alarm clock can be programmed in the following five ways:

Method	Configurable Parameters
Once a month	Date, hour, minute, second
Once a day	Hour, minute, second
Once an hour	Minute, second
Once a minute	Second
Once a second	(none)

```
     The alarm clock is set with a call to sysAlarmSet( ). This routine
     takes a method and the alarm clock parameters as arguments. The
     alarm clock can be cancelled with a call to sysAlarmCancel( ). The
     current alarm clock settings can be retrieved with a call to
     sysAlarmGet( ). The current alarm clock settings can be displayed
     with a call to sysAlarmShow( ). The routine sysAlarmIntr( ) is the
     alarm clock interrupt handler. In order to define your own
     interrupt handler, simply edit this routine.

PCI Access

     The 64-bit PCI/PCI-X busses are fully supported under the PCI
     Local Bus Specification, Revision 2.1. All configuration space
     accesses are made with BDF (bus number, device number, function
     number) format calls in the pciConfigLib module. For more
     information, refer to the man pages.

Boot Devices

     The supported boot devices are:

         sm        - shared memory (not supported in this release)
         gei0      - Primary Ethernet (10baseT, 100baseTX, or 1000baseT)
         wancom0 - Additional Ethernet (10baseT or 100baseTX)
         scsi      - SCSI

Boot Methods

     The boot methods are affected by the boot parameters. If no
     password is specified, RSH (remote shell) protocol is used. If a
     password is specified, FTP protocol is used, or, if the flag is
     set, TFTP protocol is used.

     These protocols are used for both Ethernet and shared memory boot
     devices.

ROM Considerations

     Use the following command sequence on the host to re-make the BSP
     bootrom file:

         cd target/config/mv5500
         make clean
         make bootrom
         elfToBin boot.bin
         chmod 666 boot.bin
         cp boot.bin /tftpboot/boot.bin

     Power down the board and switch ROM jumper J8 to select MOTLoad
     which normally resides in flash1 (bank B). Connect the Ethernet
     and console serial port cables, then power the board back up.

Flashing the Boot ROM Using Motorola MOTLoad:

     First set some MOTLoad global variables to conform to your
     particular operational environment. This is done via a series of
     gevEdit commands: These variables, when set, remain in NVRAM
     through power cycles and can later be changed, if desired, with
     MOTLoad gevDelete and gevEdit commands.

       MVME5500> gevEdit mot-/dev/enet1-cipa
       (Blank line terminates input.)
       123.111.32.90


       Update Global Environment Area of NVRAM (Y/N)? y

       MVME5500> gevEdit mot-/dev/enet1-sipa
       (Blank line terminates input.)
       123.111.32.180


       Update Global Environment Area of NVRAM (Y/N)? y

       MVME5500> gevEdit mot-/dev/enet1-gipa
       (Blank line terminates input.)
       123.111.32.1


       Update Global Environment Area of NVRAM (Y/N)? y
```

```
MVME5500> gevEdit mot-/dev/enet1-file
(Blank line terminates input.)
mydir/boot.bin


Update Global Environment Area of NVRAM (Y/N)? y
MVME5500>
```

The above sequence sets the client IP address (IP address of the MVME5500) to 123.111.32.90, the IP address of the server to 123.111.32.180, the IP address of the gateway to 123.111.32.1 and the tftp file name to "mydir/boot.bin". Note that we have used "dev/enet1" as the ethernet device involved in the download. You can use another device, such as "dev/enet0" if you wish, use of the netShow command will display which interfaces are "up" and available for use in the download operation.

The file is transferred from the TFTP host to the target board using the tftpGet command. IMPORTANT: You must have a TFTP server running on your host's subnet for the tftpGet command to succeed. The file name must be set to the location of the binary file on the TFTP host. The binary file must be stored in the directory identified for TFTP accesses, but the file name is a relative path and does not include the /tftpboot directory name:

Now that the MOTLoad global variables have been set into NVRAM, you can perform the tftp load of the file image with the following command:

```
MVME5500>tftpGet -d/dev/enet1
```

Notice that we have specified "/dev/enet1" as the interface. This is the same interface that was specified with the gevEdit command. If no interface is specified on the tftpGet command line, MOTLoad defaults to "dev/enet0". Also take note that you can override the NVRAM settings previously set via the gevEdit command by specifying additional MOTLoad options on the tftpGet command line. The "-c" option can override the client IP address, the "-s" option can override the server IP address, the "-g" option can override the gateway IP address, and the "-f" option can override the file name.

After the file is loaded onto the target, the flashProgram command is used to put it into soldered FLASH parts (flash0 or bank A). Before using the flashProgram command you must first determine which flash bank to specify (-d option) and the offset (-o option). To do this, first run the flashShow MOTLoad command:

```
MVME5500I> flashShow
Device-Name  Base-Address,Size  Device-Size,Count  Boot  Type
/dev/flash0  F2000000,02000000  01000000,00000002  No    Intel 28F128
/dev/flash1  FF800000,00800000  00400000,00000002  Yes   Intel 28F320
MVME5500I>
```

The above display shows that the MOTLoad was booted from the "/dev/flash1" (socketed) bank because the word "Yes" appears under the "Boot" column. Thus the other bank, namely "/dev/flash0" (labeled "No" under the "Boot" column) is available for flashing the VxWorks bootrom. The offset "-o" option parameter is computed in the following way:

```
offset_value = size - 0xfff00
```

Where "size" is 0x02000000 (taken from the "/dev/flash0" "Size" column. The value 0xfff00 is always the value used in flashing a VxWorks bootrom image. It represents 1MB minus a 0x100 offset designed to align the beginning of the image being flashed with the powerPC reset vector at 0xfff00100. Performing this computation yields:

```
offset_value = 0x02000000 - 0xfff00 = 0x01f00100
```

The flashProgram command becomes:

```
flashProgram -d/dev/flash0 -o01f00100 -nfff00
```

Because we are dealing with "/dev/flash0" which is the MOTLoad default for this command we could leave the "-d" option off but it does not hurt to include it.

Be aware that if MOTLoad is running from soldered flash ("/dev/flash0") and we wish to flash the VxWorks bootrom into the

```
     "/dev/flash1" the calculation for offset would be:

       offset_flash1 = 0x00800000 - 0xfff00 = 0x700100

     And the command would be:

       flashProgram -d/dev/flash1 -o00700100 -nfff00

IMPORTANT

     DO NOT use the flashProgram command to flash into the bank labeled
     "Yes" by the flashShow command. Doing so will destroy the MOTLoad
     firmware image.

     When the flashProgram command is finished, power down the board
     and switch ROM jumper J8 to select soldered FLASH (flash0 or bank
     A) by placing the jumper across pins 1 and 2). Power the board
     back up. The VxWorks boot image which you just flashed will be in
     control.

SPECIAL CONSIDERATIONS

     The rom resident images, bootrom_res_high, vxWorks.res_rom_res_low
     and vxWorks.res_rom_nosym_res_low mentioned in the section Make
     Targets do not work with the elfToBin tool released with Tornado
     2.0. A modified version of elfToBin has to be used in order to get
     these images working. Please refer to SPR #8845 for a new version
     of elfToBin.

Known Problems

     The following problems are known to exist with the BSP. Wind River
     has been notified of these problems.

     When through the shared memory inteface, the following warning
     message sequence appears. There are no lasting ill effects from
     this message, shared memory functions properly:

         Attached TCP/IP interface to fei unit 0
         Attaching network interface lo0... done.
         Initializing backplane net with anchor at 0x4100... done.
         Backplane anchor at 0x4100... Attaching network interface sm0... done.
         Unable to add route to 144.191.0.0; errno = 0xffffffff.

     When using the IPMC module, it is not possible to boot the kernel
     from the SCSI drive unless the following modifications are made to
     "../all/bootConfig.c":

     In the scsiLoad( ) routine remove the following section of code:

         pCbio = dpartDevCreate((CBIO_DEV_ID) pScsiBlkBootDev,
                                NUM_PARTITIONS_DISK_BOOT,
                                usrFdiskPartRead);

         if (NULL == pCbio)
             {
             printErr ("scsiLoad: dpartDevCreate returned NULL.\n");
             return (ERROR);
             }

         /* initialize the boot block device as a dosFs device named  */

         if (ERROR == dosFsDevCreate(bootDir,
                                     dpartPartGet(pCbio,PARTITION_DISK_BOOT),
                                     20, NONE))
             {
             printErr ("scsiLoad: dosFsDevCreate returned ERROR.\n");
             return (ERROR);
             }

     And replace it with:

         if (dosFsDevInit (bootDir, pScsiBlkBootDev, NULL) == NULL)
             {
             printErr ("dosFsDevInit failed.\n");
             return (ERROR);
             }

Delivered Objects

     The delivered objects are: bootrom, vxWorks, vxWorks.sym, and
     vxWorks.st.
```

Make Targets

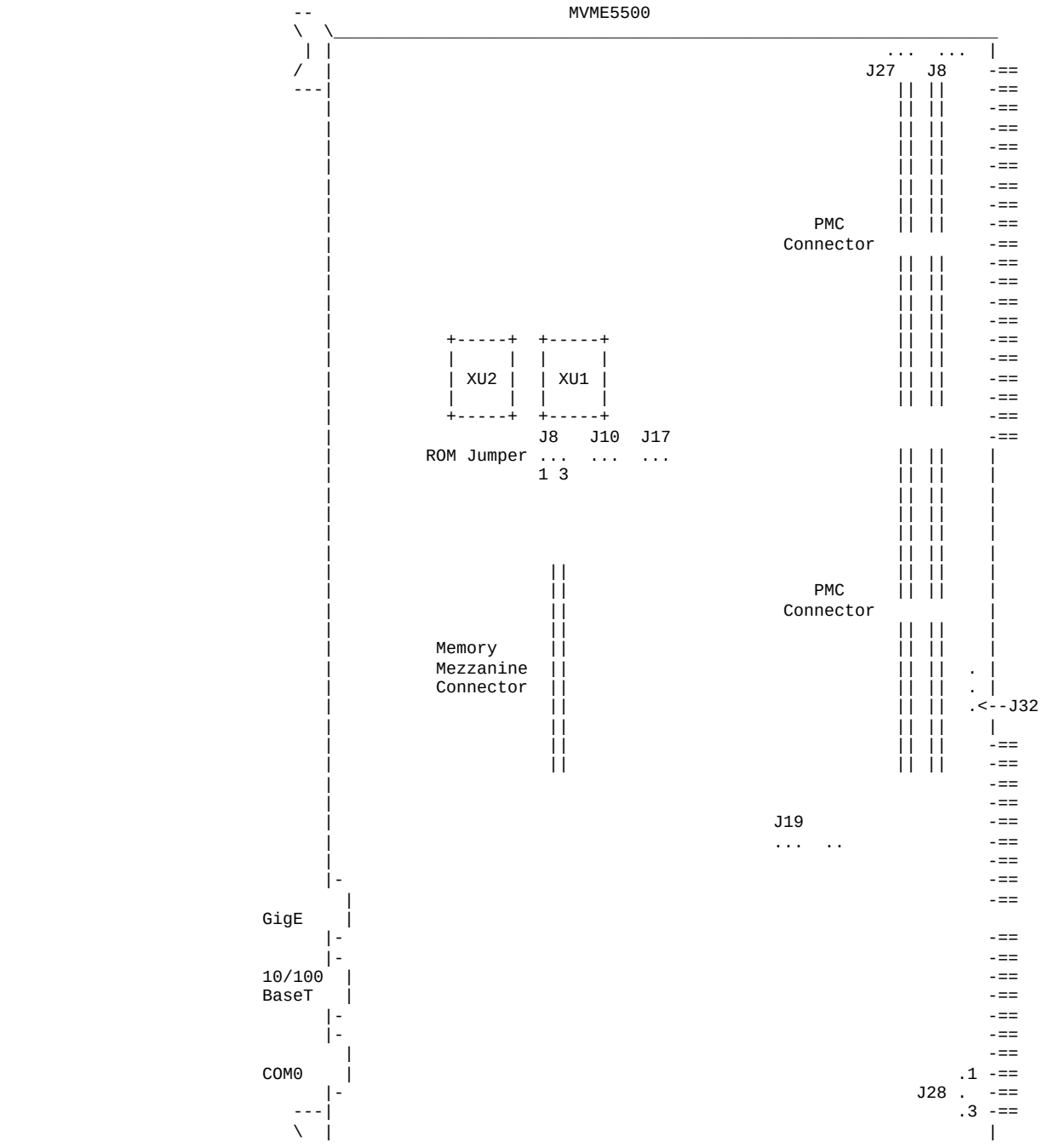
The make targets are listed as the names of object-format files.
Append .hex to each to derive a hex-format file name.

```
 bootrom
 bootrom_uncmp
 bootrom_res_high  (bootrom_res does not build)
 vxWorks (with vxWorks.sym)
 vxWorks_rom
 vxWorks.st
 vxWorks.st_rom
 vxWorks.res_rom_res_low (vxWorks.res_rom does not build)
 vxWorks.res_rom_nosym_res_low (vxWorks.res_rom_nosym does not build)
```

Special Routines

BOARD LAYOUT

The diagrams below show flash EEPROM locations and jumpers
relevant to VxWorks configuration:



```
SEE ALSO

     Tornado User's Guide: Getting Started

     VxWorks Programmer's Guide: Configuration

BIBLIOGRAPHY

     Motorola Computer Group Online Documentation

     Motorola Engineering Specification and Programming Model for
     MVME5500

     Motorola PowerPC 60X RISC Microprocessor User's Manual

     Motorola PowerPC Microprocessor Family: The Programming
     Environments

     IEEE P1386.1 Draft 2.0 - PCI Mezzanine Card Specification (PMC)

     IEEE P1386 Draft 2.0 - Common Mezzanine Card Specification (CMC)

     Motorola MPC7450 RISC Microprocessor Family User's Manual Rev 2,

     Marvell GT-64260A/B System Controller for PowerPC Processors
     Manual Rev 1.1,

     Peripheral Component Interconnect (PCI) Local Bus Specification,
     Rev 2.1
```

Downloads

Filename	Description
MVME5500_0.3.tar.gz	MVME5500
MVME5500_0.4.tar.gz	MVME5500 VxWorks BSP (1.2/0.4)